Meander's innovative curve-rendering method faithfully captures the artist's intent, resulting in a significant improvement in creative communication throughout the production pipeline.

To **Mark Rappaport** for the concept, design and development, to **Scott Oshita** for the motion analysis and CAD design, to **Jeff Cruts** for the development of the faux-hair finish techniques, and to **Todd Minobe** for the character articulation and drive-train mechanisms, of the Creature Effects Animatronic Horse Puppet.

The Animatronic Horse Puppet provides increased actor safety, close integration with live action, and improved realism for filmmakers.

To **Glenn Sanders** and **Howard Stark** for the design and engineering of the Zaxcom Digital Wireless Microphone System.

The Zaxcom system has advanced the state of wireless microphone technology by creating a fully digital modulation system with a rich feature set, which includes local recording capability within the belt pack and a wireless control scheme providing real-time transmitter control and time-code distribution.

To **David Thomas**, **Lawrence E. Fisher** and **David Bundy** for the design, development and engineering of the Lectrosonics Digital Hybrid Wireless Microphone System.

The Lectrosonics system has advanced the state of wireless microphone technology by means of an innovative digital predictive algorithm to realize full fidelity audio transmission over a conventional analog FM radio link, by reducing transmitter size, and by increasing power efficiency.

To **Parag Havaldar** for the development of expression-based facial performance-capture technology at Sony Pictures Imageworks.

This pioneering system enabled large-scale use of animation rig-based facial performance-capture for motion pictures, combining solutions for tracking, stabilization, solving and animator-controllable curve editing.

To **Nicholas Apostoloff** and **Geoff Wedig** for the design and development of animation rig-based facial performance-capture systems at ImageMovers Digital and Digital Domain.

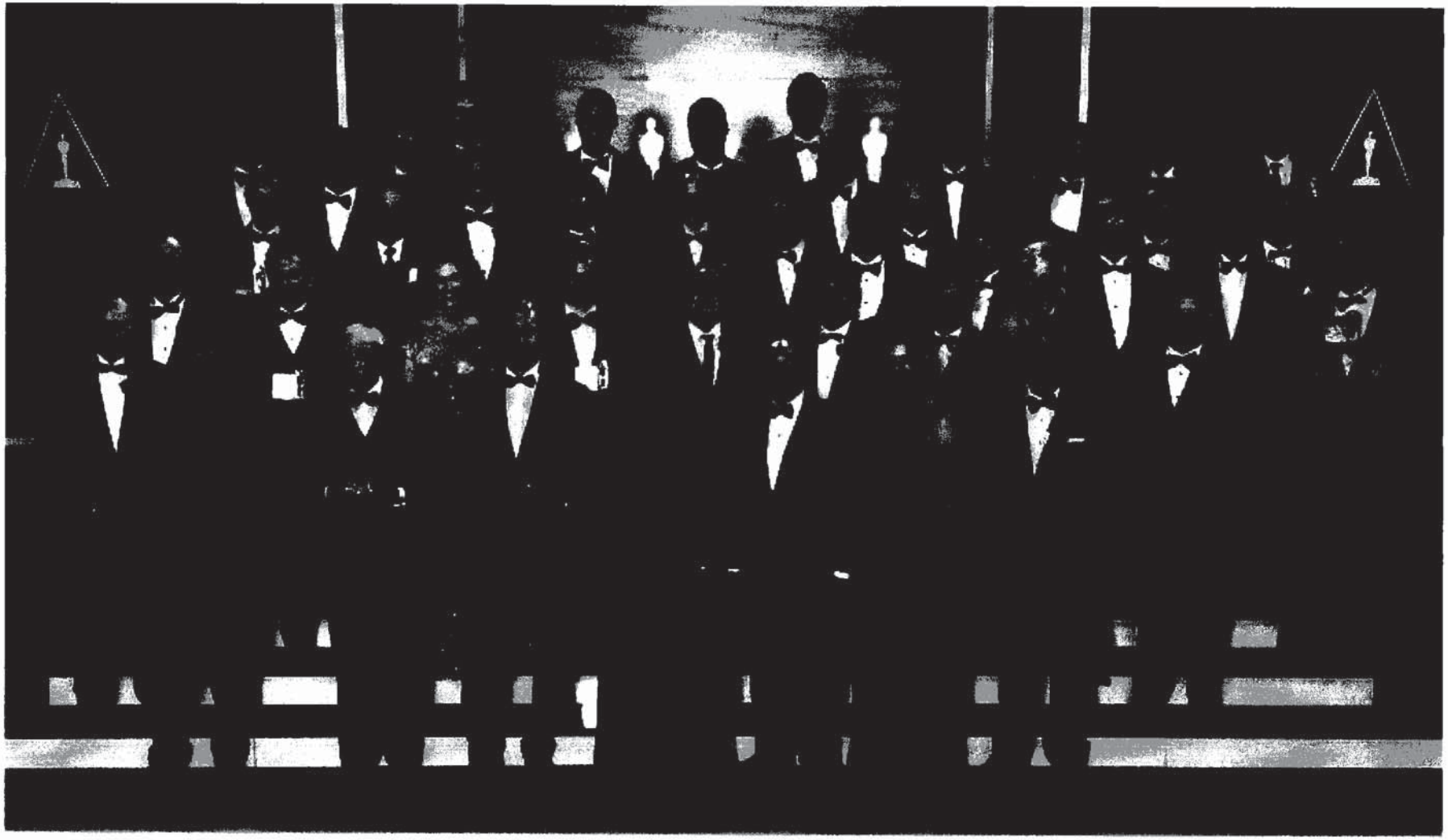
These systems evolved through independent, then combined, efforts at two different studios, resulting in an artist-controllable, editable, scalable solution for the high-fidelity transfer of facial performances to convincing digital characters.

To **Kiran Bhat**, **Michael Koperwas**, **Brian Cantwell** and **Paige Warner** for the design and development of the ILM facial performance-capture solving system.

This system enables high-fidelity facial performance transfer from actors to digital characters in large-scale productions while retaining full artistic control, and integrates stable rig-based solving and the resolution of secondary detail in a controllable pipeline.

150

EXHIBIT 129



251

13
FEB
2017

4:25 am

THE ACADEMY'S SCI-TECH AWARDS 2017 WINNERS: SEE THE FULL LIST

BY. CHARLES SCHUMLEY

The Viper camera enabled frame-based logarithmic encoding, which provided uncompressed camera output suitable for importing into existing digital intermediate workflows.

To **Larry Gritz** for the design, implementation and dissemination of Open Shading Language (OSL).

OSL is a highly optimized runtime architecture and language for programmable shading and texturing that has become a de facto industry standard. It enables artists at all levels of technical proficiency to create physically plausible materials for efficient production rendering.

To **Carl Ludwig**, **Eugene Troubetzkoy** and **Maurice van Swaaij** for the pioneering development of the CGI Studio renderer at Blue Sky Studios.

CGI Studio's groundbreaking ray-tracing and adaptive sampling techniques, coupled with streamlined artist controls, demonstrated the feasibility of ray-traced rendering for feature film production.

To **Brian Whited** for the design and development of the Meander drawing system at Walt Disney Animation Studios.

Meander's innovative curve-rendering method faithfully captures the artist's intent, resulting in a significant improvement in creative communication throughout the production pipeline.

To **Mark Rappaport** for the concept, design and development, to **Scott Oshita** for the motion analysis and CAD design, to **Jeff Cruts** for the development of the faux-hair finish techniques, and to **Todd Minobe** for the character articulation and drive-train mechanisms, of the Creature Effects Animatronic Horse Puppet.

The Animatronic Horse Puppet provides increased actor safety, close integration with live action, and improved realism for filmmakers.

To **Glenn Sanders** and **Howard Stark** for the design and engineering of the Zaxcom Digital Wireless Microphone System.

The Zaxcom system has advanced the state of wireless microphone technology by creating a fully digital modulation system with a rich feature set, which includes local recording capability within the belt pack and a wireless control scheme providing real-time transmitter control and time-code distribution.

To **David Thomas**, **Lawrence E. Fisher** and **David Bundy** for the design, development and engineering of the Lectrosonics Digital Hybrid Wireless Microphone System.

The Lectrosonics system has advanced the state of wireless microphone technology by means of an innovative digital predictive algorithm to realize full fidelity audio transmission over a conventional analog FM radio link, by reducing transmitter size, and by increasing power efficiency.

EXHIBIT 130

Business Wire
A Berkshire Hathaway Company

Autodesk

Autodesk Reveals New Gameware Advancements at GDC 2013

Game Development Solutions Boost UI Design, Artificial Intelligence, Lighting and Animation Tools

March 25, 2013 12:00 PM Eastern Daylight Time

SAN FRANCISCO--(BUSINESS WIRE)--**Game Developers Conference 2013 —** Autodesk, Inc. (NASDAQ:ADSK) unveiled new versions of its widely adopted game development middleware from the Autodesk Gameware product line that allow developers to create more compelling gaming experiences.

Updates for Autodesk Scaleform, Autodesk Navigation, Autodesk Beast and Autodesk HumanIK bring game developers enhanced compatibility for multiple platforms and game engines, increased functionality and powerful new tools. Autodesk Gameware is optimized for the current generation and certain next generation consoles.

"We are committed to helping developers of all sizes realize their creative visions. Our Gameware products allow artists to work near seamlessly across multiple platforms and engines, while removing traditional bottlenecks and accelerating production," said Chris Bradshaw, senior vice president, Media and Entertainment at Autodesk. "Our 2014 releases deliver advancements for the development community as they create new titles for both the current generation and next generation of hardware. We look forward to seeing what they come up with next."

Autodesk Scaleform 4.3 Enables Powerful UI Features Across Devices

Autodesk Scaleform 4.3 harnesses the power of the Adobe Flash toolset to help developers create immersive user interface (UI) environments while streamlining workflows and accelerating the development cycle. Used on projects ranging from blockbuster PC and console titles to social, casual and mobile games, it provides an artistic, design driven workflow to create hardware-accelerated 3D game menus, heads-up displays, animated textures, in-game videos and mini-games. With a versatile and proven toolset, Autodesk Scaleform is compatible across certain mobile platforms and is easily integrated with certain third party or in-house game engines.

New features of Scaleform 4.3 are support for additional ActionScript 3 application programming interface (API) classes to help increase platform compatibility and native extensions for Adobe AIR, which enables developers to leverage an array of third party and community-developed extensions directly in Scaleform. The latest version also allows UI designers to leverage a wider range of color and shape transformations with improved Adobe Flash blending and has files demonstrating touch and gesture UI interaction. Scaleform 4.3 is compatible with the Unity 4 engine and also supports Microsoft Windows 8. Additional improvements are enhancements to mobile device compatibility to help facilitate the deployment of powerful UI features to a wider array of devices.

Autodesk Gameware Navigation 2014 Helps Improve Artificial Intelligence

With Autodesk Gameware Navigation 2014 middleware, developers can leverage full source code access, an accessible API and new remote visual debugging tools to create complex, ambitious artificial intelligence (AI) that allows for more gameplay. The successor to Autodesk Kynapse, Navigation has automatic NavMesh generation, pathfinding and path following in complex game environments. Character and obstacle avoidance, dynamic NavMesh, swappable sectors and Unreal Engine 3 are supported out-of-the-box with Gameware Navigation, greatly improving the speed and quality of AI iteration. The Gameware Navigation API is compatible with certain major gaming platforms.

Autodesk Beast 2014 Simulates Near Realistic In-Game Lighting

Autodesk Beast 2014 middleware enables game creators to simulate natural, physically near accurate lighting effects. Using Beast, artists can design and test lighting effects in game environments faster and more efficiently – helping result in more compelling game experiences. It can be integrated into level editors and allows artists to manipulate level lighting interactively by showing a highly accurate final render preview in almost real-time. The newest version of Beast adds physics-based rendering for more natural lighting, has Open Shading Language support to help increase flexibility in creating material properties and enables more near accurate preview renders. The Autodesk Beast API is designed to make integration with many game engines simple and intuitive.

Autodesk HumanIK 2014 Augments Character Animation

Autodesk HumanIK 2014 animation middleware helps reduce the need for developers to produce large libraries of character animations, saving more time for the creative. With full-body inverse kinematics and real-time retargeting technology, the latest version of HumanIK enables characters to interact with and procedurally adapt to game environments at run-time. Optimized to help facilitate the simultaneous animation of large groups of characters, HumanIK 2014 also adds improved support for mobile platforms, has easier control for squash and stretch deformations of the neck and spine and enhances quadruped animation support. As a C++ library, HumanIK supports many major gaming platforms and helps artists bring games to life with more near realistic and immersive character animation experiences.

Availability

Scaleform 4.3, Gameware Navigation 2014, Beast 2014 and HumanIK 2014 are expected to be available in spring 2013. To learn more about Autodesk's game development tools or access evaluation versions of the products, please visit: http://gameware.autodesk.com

About Autodesk

Autodesk, Inc., is a leader in 3D design, engineering and entertainment software. Customers across the manufacturing, architecture, building, construction, and media and entertainment industries -- including the last 18 Academy Award winners for Best Visual Effects -- use Autodesk software to design, visualize and simulate their ideas. Since its introduction of AutoCAD software in 1982, Autodesk continues to develop the broadest portfolio of state-of-the-art software for global markets. For additional information about Autodesk, visit www.autodesk.com.

156

Contacts

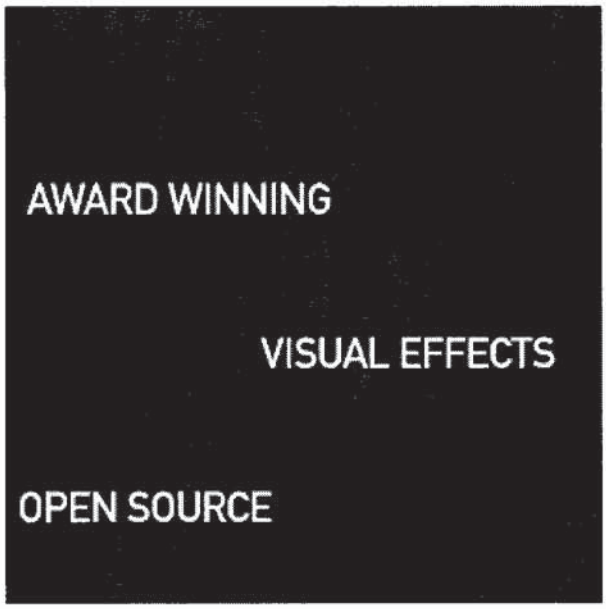
Autodesk, Inc.
Amelise Javier-Lane, 415-547-3562
amelise.javier.lane@autodesk.com
or
Karen Raz, 310-450-1482
karen@razpr.com

157

EXHIBIT 131

Imageworks Open Source

- home
- contact us
- all projects
- imageworks.com



Community: One of the promises of Open Source. We're seeing the positive effects as a community of visual effects and animation professionals come together to solve problems more effectively today than ever before. This idea of giving stuff away is catching on and our industry is benefiting.

Our projects are seeing great adoption. Alembic, one of our most ambitious collaborations to date, is supported by most major 3d applications. OpenColorIO has also been widely adopted and is helping to simplify color pipelines in tools across our industry. Open Shading Language can be found in V-Ray, Autodesk Beast, Blender Cycles and other products coming soon. We're excited.

Please take a moment to check out these and our other open source offerings. They are provided with familiar, non-restrictive open source licenses and are already in use in studios around the world. These tools have already helped Sony Pictures Imageworks put films on the screen with greater ease, and we hope they can do the same for you.

- OSL

OSL

Open Shading Language

- Alembic

Alembic

Open Interchange Format

159

EXHIBIT 132

What's New in Beast 2013.2.x

Products and versions covered

By: AUTODESK. Help
May 30 2017
| In-Product View (http://help.autodesk.com/cloudhelp/2015/ENU/Beast-SDK-Help/files/GUID-08210E19-206D-4643-96EE-24DFDEE68845.htm)
(http://help.autodesk.com/cloudhelp/2015/ENU/Beast-SDK-Help/files/GUID-08210E19-206D-4643-96EE-24DFDEE68845.htm)

What's New in Beast 2013.2.x

This page lists the new features and bug fixes introduced in the 2013.2 release.

Physically based rendering

This release introduces a totally revamped version of the Beast renderer, based around the concept of physical rendering. Some of the design goals for this new version were:

- The renderer should be easy to use, with minimal setup time for the artist.
- The live preview should match the final baking result.

- The old fixed function shading pipeline should be replaced with Open Shading Language (OSL).
- Performance should scale (almost) linearly with the number of computers you assign to your farm.

This is a major change that provides many new features, outlined below.

Physical scenes

To accommodate all the new functionality offered by the physical renderer while preserving the old functionality, this release introduces a new type of scene called a *physical scene*. This is done to ensure that you do not use the old fixed-function materials and certain old render passes with the new physical renderer.

Rendering jobs that run on physical scenes currently support the following render passes:

- Full-shading
- Illumination
- IlluminationSH
- RNM

If your project requires the physical rendering system to support for any other passes, please contact Autodesk Support. See Support (https://knowledge.autodesk.com/search-result/caas/CloudHelp/cloudhelp/2015/ENU/Beast-SDK-Help/files/GUID-D70BFD6A-2A69-4C5D-9885-D6B1F176E60A-htm.html).

Physical materials and OSL

This release completely changes the way you set up materials in your scenes. Instead of calling API functions to set up a fixed-function shader built in to Beast, you now provide an OSL shader for each material. The shader is invoked by Beast when necessary to determine the effect of light on the surface. You can use the functions provided by the Beast API to set up input parameters for the shaders for each different material you create. For details, see Creating Physical Materials (https://knowledge.autodesk.com/search-result/caas/CloudHelp/cloudhelp/2015/ENU/Beast-SDK-Help/files/GUID-B812FA2F-A188-4D9A-A5A8-ACD7A771AA89-htm.html).

Live materials and material parameters

The new physical shading system allows you to change your materials and set new parameter values for your shaders during the course of a live session. This is a major improvement over the old materials system.

eRnsT and final render convergence

When using the physically based rendering system, the same renderer is used for both eRnsT and final render. Therefore, there should be no visible artifacts introduced in the final bake step that are not visible in the preview. (This excludes artifacts that are a part of a specific pass. For example, the RNM pass might introduce anomalies that come from applying the light information onto the final format of the pass).

In addition, there are no extra settings to tweak for the final render, which used to be done in the XML file you set for the job. For example, the primary cache, secondary cache, light sample settings, etc. The properties used to configure eRnsT and final baking are the same, except for quality settings.

Note, however, that due to the use of OSL shaders, this release only supports the "illumination only" mode when running eRnsT in live bake mode.

Post-filtering

One of the major changes in this release is the removal of all cache-based GI calculations when rendering physical scenes. This makes renders much easier to set up, without the need to spend time tweaking the behavior of the different render caches through the XML configuration.

Removing the caches also allows Beast to scale better in large render farms, since most of the sync points during a render job have been eliminated.

However, since we apply a path tracer approach, the raw result will likely contain more noise than in previous versions. To counter this noise, this release also introduces a new filter that can be applied for a physical job. This filter uses more data than a regular image filter, which means that you can get fairly good results even from quite noisy input data. The filter also filters between different shapes and bake types, which means that it can smooth out both UV-seams and seams between objects.

See Creating Render Targets (https://knowledge.autodesk.com/search-result/caas/CloudHelp/cloudhelp/2015/ENU/Beast-SDK-Help/files/GUID-C5DEE2AC-1BFE-4C80-8370-03BBFD925C4D-htm.html).

Maya plug-in

The Maya plug-in has been re-engineered to always produce a physical scene. As a convenience, the plug-in transparently converts Maya materials to an OSL representation. However, for best results, and the ability to tweak materials and material settings live, you are encouraged to set up your scene with custom shaders using the new BeastOSL node. The Beast Settings node has also been updated to reflect the simplified workflow. See the Maya plug-in (https://knowledge.autodesk.com/search-result/caas/CloudHelp/cloudhelp/2015/ENU/Beast-SDK-Help/files/GUID-1A6B5739-AA7F-4155-966A-54C81585442F-htm.html) section for details.

EXHIBIT 133

nicolasleduc

602 Posts
42 Kudos
6 Solutions

Post 1 of 1

Report

New Feature: Open Shading Language Support

436 Views, 0 Replies
03-26-2013 06:07 AM

Options

This thread is to discuss the new Open Shading Language Support in Beast 2014. See a description of the feature here: Open Shading Language Support

EXHIBIT 134

Creating Physical Materials

SHARE

Materials define the way the surfaces of your meshes react to light.

You define materials within a scene. When you create a mesh, you give each of its triangles a material by name. Then, when you create an instance of a mesh within a scene, each of its triangles uses the material in that scene whose name matches the name of the material assigned to it when the mesh was created. See Creating Meshes. You can also override the materials used in each instance of a mesh. See Creating Mesh Instances.

Beast uses a physical rendering system based on shaders expressed in the Open Shading Language (OSL). To customize the way each material interacts with light, you assign it a shader. Depending on the shader you use, you may also be able to set various input parameters (colors, textures, numeric values, etc.) for the material, which are passed to the shader as input values.

NOTE: The sections on this page detail the recommended way to set up materials in Beast, using the physically based rendering system introduced in release 2013.2. For instructions on using the older, fixed-shader materials system, see Creating Classic Materials.

What is the Open Shading Language?

Open Shading Language is an open-source shading system developed by Sony Pictures Imageworks for use in major feature films.

For more information, including source code, and a complete language specification that also outlines the library of functions you can use in your shaders, see: https://github.com/imageworks/OpenShadingLanguage (https://github.com/imageworks/OpenShadingLanguage)

OSL in Beast

Since OSL was designed for VFX film production, some aspects of it are not relevant for use when baking textures for games. In particular, Beast does not support:

- Displacement Shaders
- Volumetric Shaders
- BSSDRF shaders (sub surface scattering)
- Networks of OSL shaders

If you find that you need support for these or any other unsupported aspects of OSL, please contact Autodesk Support. See Support.

Writing an OSL shader file

In order to set up a material, you need to provide a file that contains an OSL shader. A shader can be as simple or as complex as you need in order to express the way your material surface interacts with light to produce its final shading. You can use the sample shader files provided in the *shaders* sub-directory of the Beast SDK package, or write your own from scratch.

For example, the following code shows a simple shader that multiplies the diffuse *closure* in OSL with a texture file that is passed as an input parameter:

```
Surface                                                    //< All Beast shaders use the Surface type.
diffuseTexture                                             //< A name for this shader.
( string diffuseFile = ""                                  //< An input parameter.
     [[  string description = "Diffuse texture file" ]] )  //< Anything inside [[ ]] is optional metadata.
{
    Ci = (color)texture (filename, u, v) * diffuse(N);     //< Beast expects the shader to set the output
                                                           //< closure in the Ci variable.
}
```

For a more complex and interesting example, see the `beastphong.osl` file, which models a Phong-type shading similar to the one used by Beast to render "classic" non-physical materials (see also Creating Classic Materials).

Setting up a physical material

Once you have your shader file ready, you need to:

1. Create a new `ILBMaterialHandle` .
2. Initialize the handle by calling `ILBCreateMaterial()` . In your call, you have to specify the handle of the scene that will contain your material, and the name of the material. This name should match the name of the material assigned to the triangles in your mesh.
3. Create a new `ILBShaderHandle` .
4. Initialize the shader by calling `ILBCreateShader()` . In your call, you have to specify the handle of the scene that will contain your shader, and the path to your shader file.
5. Assign your shader to your material by calling `ILBSetShader()` .
6. Set up any input parameters needed by the shader. See Creating Physical Materials below.

```
// Create and initialize the material.
ILBMaterialHandle material;
ILBCreateMaterial(scene, _T("materialWithShader"), &material);

// Create and initialize the shader.
ILBShaderHandle diffuseShader;
ILBCreateShader(scene, _T("diffuse"), _T("diffuseTexture.osl"), &diffuseShader);

// Assign the shader to the material.
ILBSetShader(material, diffuseShader);

// Create a new texture, and bind it to the material as an input parameter for the shader.
// Note that the call to ILBSetShaderParamTexture() specifies the name "diffuseFile", which is the
// name of the input parameter in the simple shader file shown above.
ILBTextureHandle tex;
ILBReferenceTexture(manager, _T("AUniqueName"), _T("C:\textures\wood.exr"), &tex);
ILBSetShaderParamTexture(material, _T("diffuseFile"), tex);
```

Note that this is a very simple example. A more in-depth usage might involve iterating over the attributes accepted by the shader, querying the type of each attribute (float, color, texture, etc.), and binding the necessary values to the material.

For a more complex example, look at the source for the Maya plugin. A Python script is used to extract the attributes accepted by the shader. These attributes are used to set up the BeastOSL node in Maya. The user can set values in the Maya UI. The settings in the Maya node are then queried, parsed, and bound to the material using that shader. See the `createMaterial()` method in `scenemanager.cpp` .

Binding input parameters

You can use the Beast API to set the following types of input parameters for your shaders:

- Floating-point numbers, using `ILBSetShaderParamFloat()` .
- Integers, using `ILBSetShaderParamInt()` .
- Textures, using `ILBSetShaderParamTexture()` .
- RGB colors, using `ILBSetShaderParamColor()` .
- UV sets, using `ILBSetShaderParamUV()` .

Each of these functions requires a material handle, a string that matches the name of the corresponding input parameter declared in the shader file, and the value that you want to pass to the shader.

Note that the input parameters are bound to a *material* handle, rather than to the *shader* handle. This allows you to use the same shader for multiple different kinds of materials, but to specify different input parameters for each of the different materials.

For example, in the `examples-physical` project, two different materials are created using the same shader, and set up with different parameters:

```
ILBShaderHandle phongishShader;
bex::apiCall(ILBCreateShader(bmh, _T("PhongishShader"), "../../data/phongish.osl", &phongishShader));
...

// Sphere 1 - Textured Lambert
ILBMaterialHandle sm1;
bex::apiCall(ILBCreateMaterial(scene, _T("TexturedLambert"), &sm1));
bex::apiCall(ILBSetShader(sm1, phongishShader));
bex::apiCall(ILBSetShaderParamTexture(sm1, _T("DiffuseTexture"), xorTexture));
bex::apiCall(ILBSetShaderParamColor(sm1, _T("DiffuseColor"), &ILBLinearRGB(0.5f, 0.5f, 1.0f)));
bex::apiCall(ILBSetMaterialOverrides(sphereInstances[1], &sm1, 1));

// Sphere 2 - Phongish
ILBMaterialHandle sm2;
bex::apiCall(ILBCreateMaterial(scene, _T("Phongish"), &sm2));
bex::apiCall(ILBSetShader(sm2, phongishShader));
bex::apiCall(ILBSetShaderParamColor(sm2, _T("DiffuseColor"), &ILBLinearRGB(0.0f, 0.0f, 0.0f)));
bex::apiCall(ILBSetShaderParamColor(sm2, _T("SpecularColor"), &ILBLinearRGB(1.0f, 0.0f, 0.0f)));
bex::apiCall(ILBSetShaderParamColor(sm2, _T("Shininess"), &ILBLinearRGB(400.0f,400.0f,400.0f)));
bex::apiCall(ILBSetMaterialOverrides(sphereInstances[2], &sm2, 1));
```

171

Despite using the same shader, the result of rendering the two spheres is very different.

Thread safety

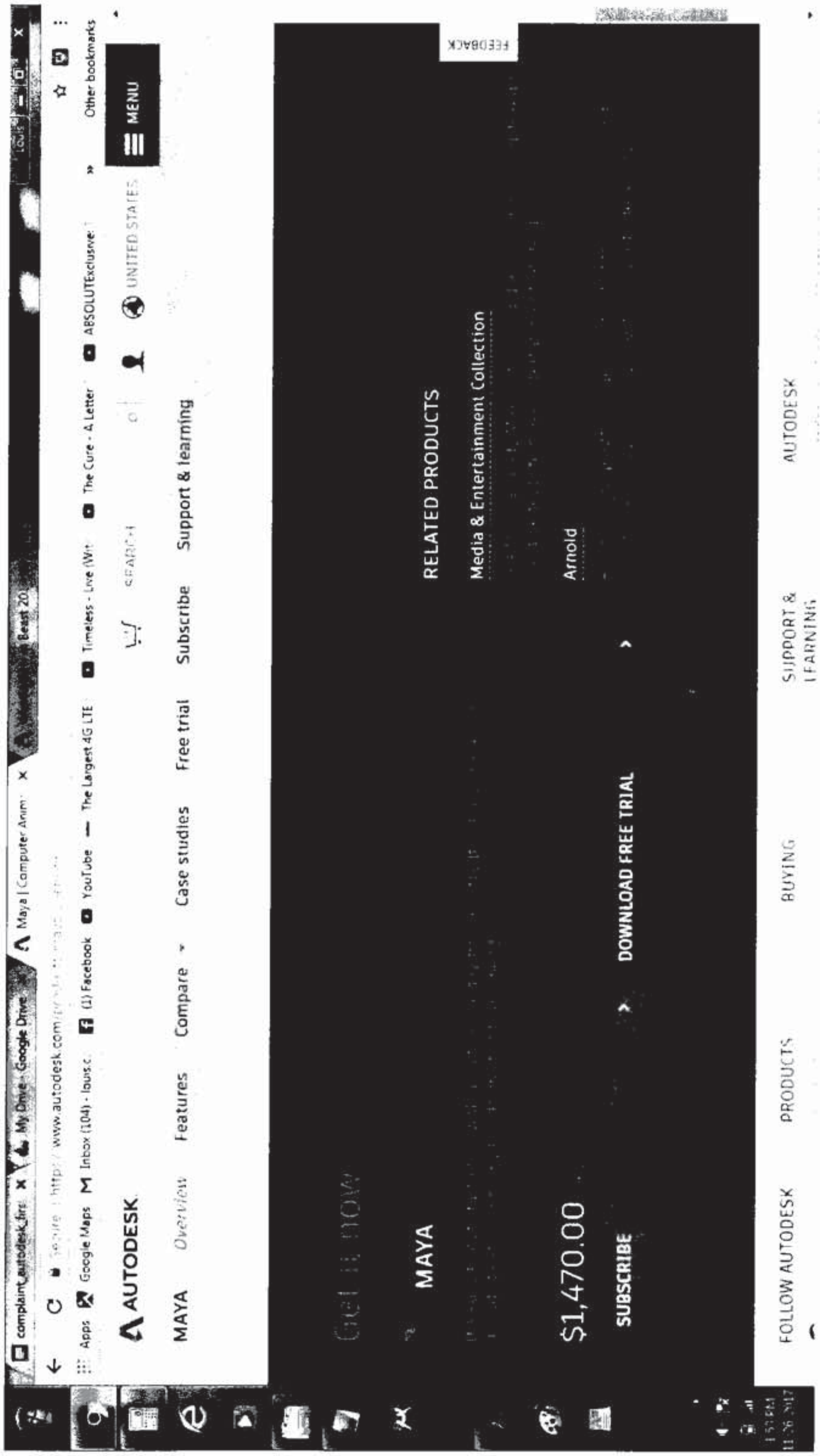
You can create multiple different materials simultaneously in multiple threads. In addition, multiple threads can find materials in the cache and use them simultaneously. You can also modify a single material from multiple different threads at the same time.

Related API functions

API functions related to the creation and setup of materials are declared in the `beastmaterial.h` file.

Topics in this section

EXHIBIT 135


AUTODESK
MAYA
Overview
Features
Compare
Case studies
Free trial
Subscribe
Support & learning
UNITED STATES
MENU
FEEDBACK
MAYA
$1,470.00
SUBSCRIBE
DOWNLOAD FREE TRIAL
RELATED PRODUCTS
Media & Entertainment Collection
Arnold
FOLLOW AUTODESK
PRODUCTS
BUYING
SUPPORT & LEARNING
AUTODESK

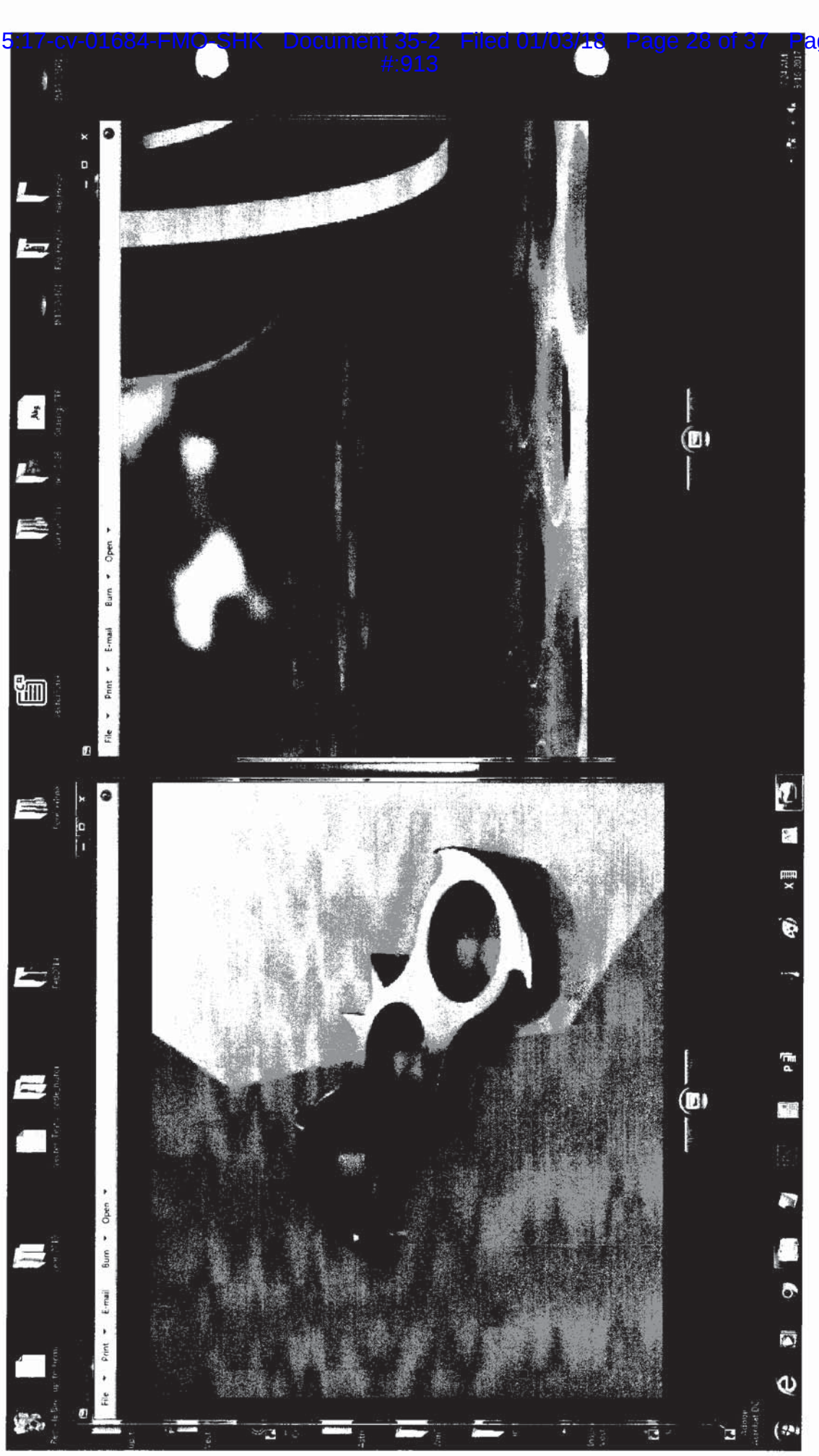
173

EXHIBIT 136

COFFELT

OSL

EXHIBIT 137



COFFELT

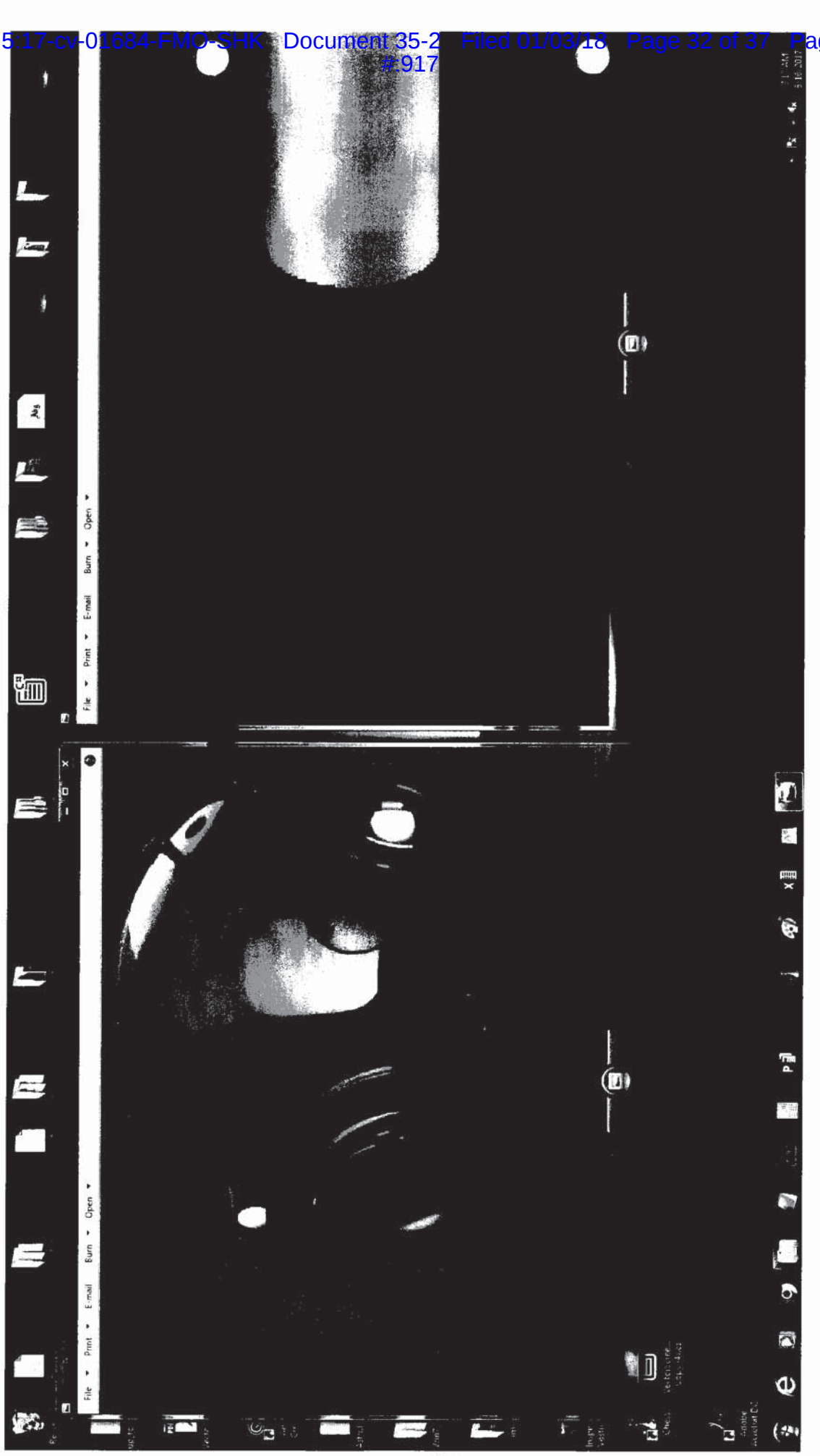
OSL

177

EXHIBIT 138

COFFELT 150

EXHIBIT 139

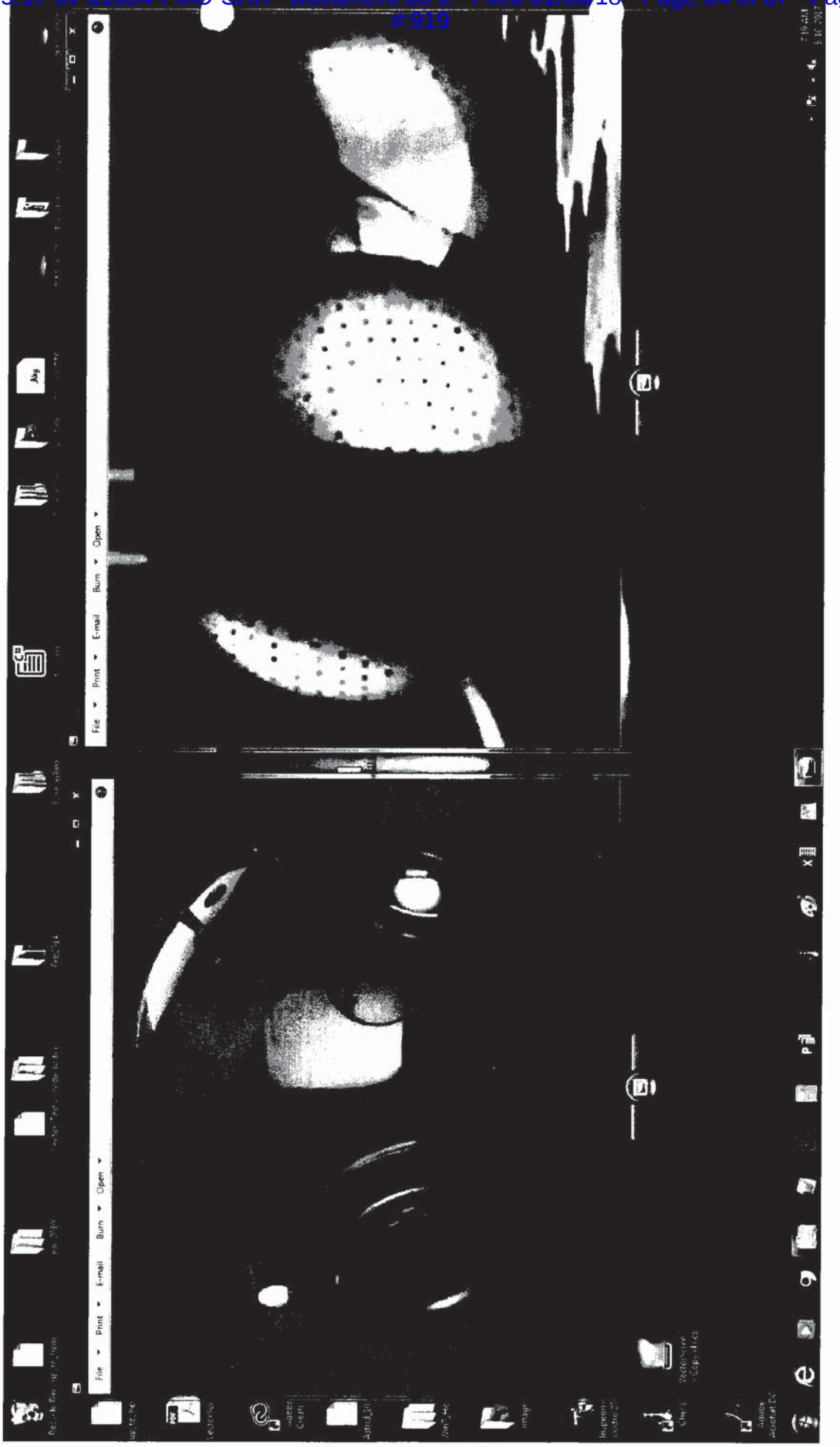


COFFELT

OSL

181

EXHIBIT 140

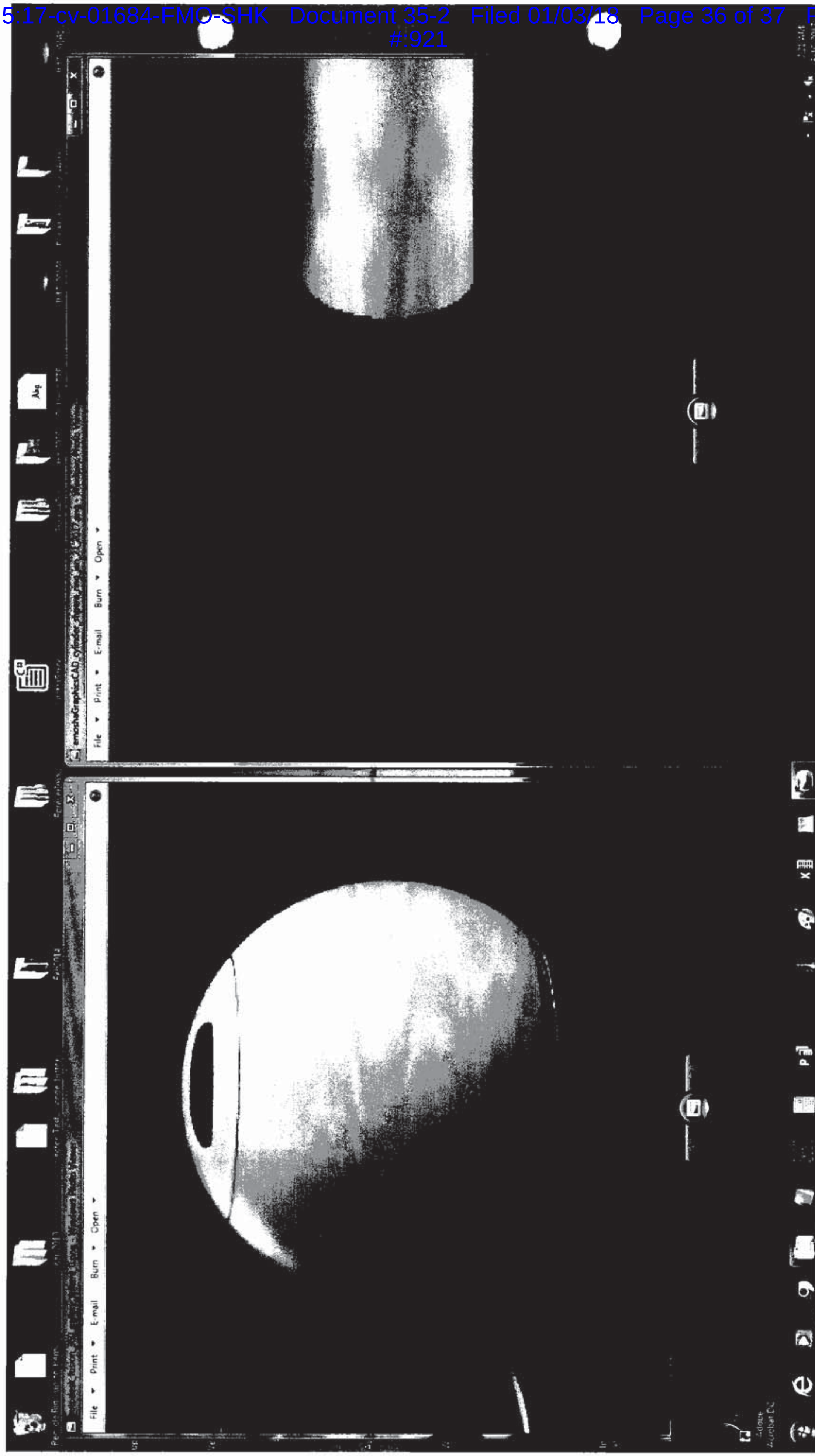


COFFELT

OSL

183

EXHIBIT 141



COFFELT

OSL

CERTIFICATE OF SERVICE

I, Louis A. Coffelt, Jr. ("Coffelt") hereby certify that on the 18th day of December, 2017, Coffelt filed the the foregoing document **SECOND AMENDED COMPLAINT FOR COPYRIGHT INFRINGEMENT** with the Clerk of the Court, in case No. 5:17-cv-01684-FMO-SHK, as follows:

Office of the Clerk
United States District Court for the Central District of California
350 W. 1st Street, Los Angeles, CA 90012

And further caused to be served one copy of the foregoing document **SECOND AMENDED COMPLAINT FOR COPYRIGHT INFRINGEMENT** by U.S. mail, postage prepaid, in case 5:17-cv-01684-FMO-SHK to the following attorneys of record:

RICHARD S.J. HUNG
rhung@mofo.com
Morrison & Foerster LLP
425 Market Street
San Francisco, CA 94105
(415) 268-7602 (direct)
(415) 268-7522 (fax)

Date: December 18, 2017

By: Louis A. Coffelt, Jr.

Louis A. Coffelt, Jr.

Plaintiff

Pro Se